Fuzzy Preferences in Multi-Objective Optimization (MOO)

The present invention relates to a method for the optimization of multi-objective problems using evolutionary algorithms, to the use of such a method for the optimization of aerodynamic or hydrodynamic bodies as well as to a computer software program product for implementing such a method.

The background of the present invention is the field of evolution algorithms. Therefore, with reference to Fig. 1, at first the known cycle of an evolutionary algorithm will be explained.

In a step S1, the object parameters to be optimized are encoded in a string called 'individual'. A plurality of such individuals comprising the initial parent generation is then generated and the quality (fitness) of each individual in the parent generation is evaluated. In a step S2, the parents are reproduced by applying genetic operators called mutation and recombination. Thus, a new generation is reproduced in step S3, which is called the offspring generation. The quality of the offspring individuals is evaluated using a fitness function that is the objective of the optimization in step S4. Finally, depending on the calculated quality value, step S5 selects, possibly stochastically, the best offspring individuals (survival of the fittest) which are used as parents for the next generation cycle if the termination condition in step S6 is not satisfied.

Before evaluating the quality of each individual, decoding may be needed depending on the encoding scheme used in the evolutionary algorithm. It should be noted that the steps S2,

The algorithm of this evolutionary optimization can be expressed by the following pseudo-code:

```
t := 0
encode and initialize P(0)
decode and evaluate P(0)
do
    recombine P(t)
    mutate P(t)
    decode P(t)
    evaluate P(t)
    P(t+1) := select P(t)
    encode P(t+1)
    t := t + 1
until terminate
```

Thereby,

- $P(0)$ denotes the initial population size ($t = 0$),
- $P(t)$ denotes the offspring population size in the t-th successor generation ($t > 0$),
- t is the index for the generation number ($t \in \mathbb{N}_0$).

Such evolutionary algorithms are known to be robust optimizers that are well-suited for discontinuous and multi-modal objective functions. Therefore, evolutionary algorithms have successfully been applied e.g. to mechanical and aerodynamic optimization problems, including preliminary turbine design, turbine blade design, multi-disciplinary rotor blade design, multi-disciplinary wing platform design and a military airframe preliminary design.

For example, details on evolutionary algorithms can be found in "Evolutionary Algorithms in Engineering Applications" (Springer-Verlag, 1997) by Dasgupta et al., and "Evolutionary Algorithms in Engineering and Computer Science" (John Wiley and Sons, 1999) by Miettinnen et al.

In the framework of the present invention, the evolutionary algorithms are applied to the simultaneous optimization of multiple objectives, which is a typical feature of practical engineering and design problems. The principle multi-objective optimization differs from that in a single-objective optimization. In single-objective optimization, the target is to find the best design solution, which corresponds to the minimum or maximum value of the objective function. On the contrary, in a multi-objective optimization with conflicting objectives, there is no single optimal solution. The interaction among different objectives gives rise to a set of compromise solutions known as the Pareto-optimal solutions. A definition of 'Pareto-optimal' and 'Pareto front' can be found in "Multi-Objective Evolutionary Algorithms: Analyzing the State of the Art" (Evolutionary Computation, 8(2), pp. 125-147, 2000) by D.A. Van Veldheizen and G. B. Lamont.

Since none of these Pareto-optimal solutions can be identified as better than others without any further consideration, the target in a multi-objective optimization is to find as many Pareto-optimal solutions as possible. Once such solutions are found, it usually requires a higher-level decisionmaking with other considerations to choose one of them for implementation.

Usually, there are two targets in a multi-objective optimization:

(i) finding solutions close to the true Pareto-optimal solutions, and

(ii) finding solutions that are widely different from each other.

The first task is desired to satisfy optimality conditions in the obtained solutions. The second task is desired to have no bias towards any particular objective function.

In dealing with multi-objective optimization problems, classical search and optimization methods are not efficient, simply because

- most of them cannot find multiple solutions in a single run, thereby requiring them to be applied as many times as the number of desired Pareto-optimal solutions,

- multiple application of these methods do not guarantee finding widely different Pareto-optimal solutions, and

- most of them cannot efficiently handle problems with discrete variables and problems having multiple optimal solutions.

On the contrary, the studies on evolutionary search algorithms, over the past few years, have shown that these methods can efficiently be used to eliminate most of the difficulties of classical methods mentioned above. Since they use a population of solutions in their search, multiple Pareto-optimal solutions can, in principle, be found in one

single run. The use of diversity-preserving mechanisms can be added to the evolutionary search algorithms to find widely different Pareto-optimal solutions.

A large number of evolutionary multi-objective algorithms (EMOA) have been proposed. So far, there are three main approaches to evolutionary multi-objective optimization, namely, aggregation approaches, population-based non-Pareto approaches and Pareto-based approaches. In the recent years, the Pareto-based approaches have been gaining increasing attention in the evolutionary computation community and several successful algorithms have been proposed. Unfortunately, the Pareto-based approaches are often very time-consuming.

Despite their shortcomings, weighted aggregation approaches to multi-objective optimization according to the state of the art are very easy to implement and computationally efficient. Usually, aggregation approaches can provide only one Pareto-solution if the weights are fixed using problem-specific prior knowledge. However, it is also possible to find more than one Pareto solution using this method by changing the weights during optimization. The weights of the different objectives are encoded in the chromosome to obtain more than one Pareto solutions. Phenotypic fitness sharing is used to keep the diversity of the weight combinations and mating restrictions are required so that the algorithm can work properly.

It has been found that the shortcomings of the conventional aggregation approach can be overcome by systematically changing the weights during optimization without any loss of simplicity and efficiency. Three methods have been proposed to change the weights during optimization to approximate the Pareto front. The randomly-weighted aggregation (RWA) method

dividuals within the population and the weights are re-distributed in each generation. In contrast, the dynamically-weighted aggregation (DWA) method changes the weights gradually when the evolution proceeds. If the Pareto-optimal front is concave, the bang-bang weighted aggregation (BWA) can also be used. In order to incorporate preferences, both RWA and DWA can be used.

Randomly Weighted Aggregation

In the framework of evolutionary optimization it is natural to take advantage of the population for obtaining multiple Pareto-optimal solutions in one run of the optimization. On the assumption that the i-th individual in the population has its own weight combination $\left(w_1^i(t),\ w_2^i(t)\right)$ in generation t, the evolutionary algorithm will be able to find different Pareto-optimal solutions. To realize this, it can be found that the weight combinations need to be distributed uniformly and randomly among the individuals, and a re-distribution is necessary in each generation:

$$w_1^i(t) = \frac{rdm\,(P)}{P},$$

$$w_2^i(t) = 1.0 - w_1^i(t),$$

wherein

- i denotes the i-th individual in the population ($i = 1, 2, \ldots, P$),
- P is the population size ($P \in \mathbb{N}$), and
- t is the index for the generation number ($t \in \mathbb{N}_0$).

The function $rdm\,(P)$ generates a uniformly distributed random number between 0 and P. In this way, a uniformly distributed random weight combination $\left(w_1^i,\ w_2^i\right)$ among the individuals can

random weight combination (w_1^i, w_2^i) among the individuals can be obtained, where $0 \leq w_1^i, w_2^i \leq 1$ and $w_1^i + w_2^i = 1$. In this context, it should be noted that the weight combinations are regenerated in every generation.

Dynamic Weighted Aggregation

In the dynamically-weighted aggregation (DWA) approach, all individuals have the same weight combination, which is changed gradually generation by generation. Once the individuals reach any point on the Pareto front, the slow change of the weights will force the individuals to keep moving gradually along the Pareto front if the Pareto front is convex. If the Pareto front is concave, the individuals will still traverse along the Pareto front, however, in a different fashion. The change of the weights can be realized as follows:

$$w_1(t) = |\sin(2\pi t/F)|,$$
$$w_2(t) = 1.0 - w_1(t).$$

where t is the number of generation. Here the sine function is used simply because it is a plain periodical function between 0 and 1. In this case, the weights $w_1(t)$ and $w_2(t)$ will change from 0 to 1 periodically from generation to generation. The change frequency can be adjusted by F. The frequency should not be too high so that the algorithm is able to converge to a solution on the Pareto front. On the other hand, it seems reasonable to let the weight change from 0 to 1 at least twice during the whole optimization.

In the above methods, it is assumed that all objectives are of the same importance. In this case, weights are changed between [0,1] in RWA and DWA to achieve all Pareto-optimal

solutions. However, in many real-world applications, different objectives may have different importance. Thus, the goal is not to get the whole Pareto front, but only the desired part of the Pareto front. The importance of each objective is usually specified by the human user in term of preferences. For example, for a two-objective problem, the user may believe that one objective is more important than the other. To achieve the desired Pareto-optimal solutions, preferences need to be incorporated into multi-objective optimization. Instead of changing the weights between [0,1], they are changed between $[w^{min}, w^{max}]$, where $0 \leq w^{min} < w^{max} \leq 1$, are defined by the preferences. Usually, the preferences can be incorporated before, during or after optimization. In this invention, preference incorporation before optimization is concerned.

As discussed in "Use of Preferences for GA-based Multi-Objective Optimization" (Proceedings of 1999 Genetic and Evolutionary Computation Conference, pp. 1504-1510, 1999) by Cvetkovic et al., the incorporation of fuzzy preferences before optimization can be realized in two ways:

- Weighted Sum: Use of the preferences as *a priori* knowledge to determine the weight for each objective, then direct application of the weights to sum up the objectives to a scalar. In this case, only one solution will be obtained.

- Weighted Pareto Method: The non-fuzzy weight is used to define a weighted Pareto non-dominance:

$$U \succeq_w V \text{ if and only if } \frac{1}{k}\sum_{i=1}^{k} w_i I_{\geq}(u_i, v_i) \geq 1$$

with the utility sets

$$U := \{u_i \mid i = 1, 2, 3, \ldots, k\} \text{ for } u_i \in [0,1] \text{ and}$$
$$V := \{v_i \mid i = 1, 2, 3, \ldots, k\} \text{ for } v_i \in [0,1],$$

where

$$I_2(u_i, v_i) = \begin{cases} 1 \text{ for } u_i \geq v_i \\ 0 \text{ for } u_i < v_i \end{cases} \text{ and } \sum_{i=1}^{k} w_i = 1.$$

A general procedure for applying fuzzy preferences to multi-objective optimization is illustrated in Fig. 2. It can be seen that before the preferences can be applied in MOO, they have to be converted into crisp weights first. The procedure of conversion is described as follows:

Given L experts (with the indices $m = 1,2,\ldots,L$) and their preference relation $\underline{\underline{P}}^m$, where $\underline{\underline{P}}^m$ is a $(k \times k)$-matrix with p_{ij} denoting the linguistic preference of the objective o_i over the objective o_j (with the indices $i,j = 1,2,\ldots,k$). Then, based on the group decision-making method, they can be combined into a single collective preference $\underline{\underline{P}}^c$. Each element of said preference matrix $\underline{\underline{P}}^c$ is defined by one of the following linguistic terms:

"much more important" (MMI),
"more important" (MI),
"equally important" (EI),
"less important" (LI), and
"much less important" (MLI).

For the sake of simplicity, the superscript c indicating the collective preference is omitted in the following text. Before converting the linguistic terms into real-valued

weights, they should at first be converted into numeric preferences. To this end, it is necessary to use the following evaluations, to replace the linguistic preferences p_{ij} in the preference matrix, as indicated in "Use of Preferences for GA-based Multi-Objective Optimization" (Proceedings of 1999 Genetic and Evolutionary Computation Conference, pp. 1504-1510) by Cvetkovic et al.

$$\begin{aligned} a \text{ is } \textit{much less important} \text{ than } b &\Rightarrow p_{ij} = \alpha,\ p_{ji} = \beta, \\ a \text{ is } \textit{less important} \text{ than } b &\Rightarrow p_{ij} = \gamma,\ p_{ji} = \delta, \\ a \text{ is } \textit{equally important} \text{ as } b &\Rightarrow p_{ij} = \varepsilon,\ p_{ji} = \varepsilon. \end{aligned}$$

The value of the parameters needs to be assigned by the decision-making and the following conditions should be satisfied in order not to lose the interpretability of the linguistic terms:

$$\alpha < \gamma < \varepsilon = 0.5 < \delta < \beta,$$
$$\alpha + \beta = 1 = \gamma + \delta.$$

Consider an MOO problem with six objectives $\{o_1, o_2, \ldots, o_6\}$ as used in "Use of Preferences for GA-based Multi-Objective Optimization" (Proceedings of 1999 Genetic and Evolutionary Computation Conference, pp. 1504-1510) by Cvetkovic et al. Suppose that among these six objectives o_1 and o_2, o_3 and o_4 are *equally important*. Thus, there are have four classes of objectives:

$$c_1 := \{o_1, o_2\},\ c_2 := \{o_3, o_4\},\ c_3 := \{o_5\} \text{ and } c_4 := \{o_6\}.$$

Besides, there are the following preference relations:

c_1 is *much more important* than c_2;
c_1 is *more important* than c_3;

c_4 is *more important* than c_1;

c_3 is *much more important* than c_2.

From these preferences, it is easy to get the following preference matrix:

$$\underline{\underline{P}} = \begin{pmatrix} EI & MMI & MI & LI \\ MLI & EI & MLI & MLI \\ LI & MMI & EI & LI \\ MI & MMI & MI & EI \end{pmatrix}.$$

From the above fuzzy preference matrix, the following real-valued preference relation matrix $\underline{\underline{R}}$ are obtained:

$$\underline{\underline{R}} = \begin{pmatrix} \varepsilon & \beta & \delta & \gamma \\ \alpha & \varepsilon & \alpha & \alpha \\ \gamma & \beta & \varepsilon & \gamma \\ \delta & \beta & \delta & \varepsilon \end{pmatrix}.$$

Based on this relation matrix, the weight for each objective can be obtained by:

$$w(o_i) = \frac{S\left(o_i, \underline{\underline{R}}\right)}{\sum_{i=1}^{k} S\left(o_i, \underline{\underline{R}}\right)}$$

with

$$S\left(o_i, \underline{\underline{R}}\right) := \sum_{j=1, j \neq i}^{k} p_{ij}.$$

For the above example, this results in

$$w_1 = w_2 = \frac{2 - \alpha}{8 + 2\alpha},$$

$$w_3 = w_4 = \frac{3\alpha}{8 + 2\alpha},$$

$$w_5 = \frac{1 - \alpha + 2\gamma}{8 + 2\alpha}, \text{ and}$$

$$w_6 = \frac{3 - \alpha - 2\gamma}{8 + 2\alpha}.$$

Since α and γ can vary between 0 and 0.5, one needs to heuristically specify a value for α and γ (recall that $\alpha < \gamma$) to convert the fuzzy preferences into a single-valued weight combination, which can then be applied to a conventional weighted aggregation to achieve one solution.

In order to convert fuzzy preferences into one weight combination, it is necessary to specify a value for α and γ. On the one hand, there are no explicit rules on how to specify these parameters, on the other hand, a lot of information will be lost in this process.

In view of this disadvantage it is the target of the present invention to improve the use of fuzzy preferences for multi-objective optimization.

This target is achieved by means of the features of the independent claims. The dependent claims develop further the central idea of the present invention.

According to the main aspect of the invention, e.g. fuzzy preferences are converted into a weight combination with each weight being described by an interval instead of a single value.

Further objects, advantages and features of the invention will become evident for the man skilled in the art when reading the following detailed description of the invention and by reference to the figures of the enclosed drawings.

Fig. 1	shows a cycle of an evolution strategy,
Fig. 2	shows schematically a procedure to apply fuzzy preferences in MOO,
Figs. 3a, 3b	show the change of weights (w_1 and w_2) with the change of parameter (α), respectively, and
Figs. 4a, 4b	show the change of weights (w_3 and w_4) with the change of parameters (α and γ), respectively.

According to the underlying invention, linguistic fuzzy preferences can be converted into a weight combination with each weight being described by an interval.

Figs. 3a, 3b, 4a, 4b show how the value of the parameters affects that of the weights. It can be seen from these figures that the weights vary a lot when the parameters (α, γ) change in the allowed range. Thus, each weight obtained from the fuzzy preferences is an interval on [0,1]. Very interestingly, a weight combination in interval values can nicely be incorporated into a multi-objective optimization with the help of the RWA and DWA, which is explained e.g. in "Evolutionary Weighted Aggregation: Why does it Work and How?" (in: Proceedings of Genetic and Evolutionary Computation Conference, pp. 1042-1049, 2001) by Jin et al.

On the assumption that the maximal and minimal value of a weight are w_{max} and w_{min} when the parameters change, the weights are changed during an optimization algorithm in the following form, which is extended from RWA:

$$w_1^i(t) = w_1^{min} + \left(w_1^{max} - w_1^{min}\right) \cdot \frac{rdm\,(P)}{P},$$

where t is the generation index. Similarly, by extending the DWA, the weights can also be changed in the following form to find out the preferred Pareto solutions:

$$w_1^i(t) = w_1^{min} + \left(w_1^{max} - w_1^{min}\right) \cdot \left| sin\left(2\,\pi t/F\right)\right|,$$

where t is the generation index. In this way, the evolutionary algorithm is able to provide a set of Pareto solutions that are reflected by the fuzzy preferences. However, it is recalled that DWA is not able to control the movement of the individuals if the Pareto front is concave, therefore, fuzzy preferences incorporation into MOO using DWA is applicable to convex Pareto fronts only, whereas the RWA method is applicable to both convex and concave fronts.

To illustrate the underlying invention, some examples on two-objective optimization using the RWA are presented in the following. In the simulations, two different fuzzy preferences are considered:

1. *Objective 1 is more important than objective 2;*
2. *Objective 1 is less important than objective 2.*

For the first preference, one obtains the following preference matrix:

$$\underline{\underline{P}} = \begin{pmatrix} 0.5 & \delta \\ \gamma & 0.5 \end{pmatrix},$$

with $0.5 < \delta < 1$ and $0 < \gamma < 0.5$. Therefore, the weights for the two objectives using the RWA method are:

$$w_1^i(t) = 0.5 + 0.5 \cdot \frac{rdm(P)}{P},$$

$$w_2^i(t) = 1.0 - w_1^i(t).$$

Similarly, the following weights are obtained for the second preference:

$$w_1^i(t) = 0 + 0.5 \cdot \frac{rdm(P)}{P},$$

$$w_2^i(t) = 1.0 - w_1^i(t).$$

To summarize, the invention proposes a method to obtain the Pareto-optimal solutions that are specified by human preferences. The main idea is to convert the fuzzy preferences into interval-based weights. With the help of the RWA and DWA, it is shown to be successful to find the preferred solutions on two test functions with a convex Pareto front. Compared to the method described in "Use of Preferences for GA-based Multi-Objective Optimization" (Proceedings of 1999 Genetic and Evolutionary Computation Conference, pp. 1504-1510, 1999) by Cvetkovic et al., the method according to the invention is able to find a number of solutions instead of only one, given a set of fuzzy preferences over different objectives. This is consistent with the motivation of fuzzy logic.

Many technical, industrial and business applications are possible for evolutionary optimization. Examples for applications can be found e.g. in "Evolutionary Algorithms in Engineering Applications" (Springer-Verlag, 1997) by Dasgupta et al., and "Evolutionary Algorithms in Engineering and

Computer Science" (John Wiley and Sons, 1999) by Miettinnen et al.